occam 2 toolset handbook

INMOS Limited

Contents

The occam 2 Toolset

Tool	Description
`icollect`	The code collector which creates bootable files from linked units or configuration binary files.
`icvemit`	Memory interface file convertor. Converts files produced by `iemi` (a previous version of `iemit`) to a format suitable for use with `ieprom` and `iemit`.
`icvlink`	The TCOFF file convertor which converts LFF object files to TCOFF format.
`idebug`	The network debugger which provides post-mortem and interactive debugging of transputer programs.
`idump`	The memory dumper for saving the program image on the root transputer.
`iemit`	The transputer memory configurer for evaluating and defining memory configurations.
`ieprom`	The EPROM program formatter tool which generates transputer bootable code for input to ROM programmers.
`ilibr`	The toolset librarian which builds libraries of compiled code.
`ilink`	The toolset linker which links separately compiled code into a single file.
`ilist`	The binary lister which displays information from toolset object files.
`imakef`	The Makefile generator which generates Makefiles for input to MAKE programs.
`iserver`	The host file server which loads programs onto transputer hardware and provides host communication.
`isim`	The T425 simulator. Simulates program execution on an IMS T425 transputer.
`iskip`	The skip loader tool which loads over the root transputer.
`oc`	The occam 2 compiler. Generates object code for specific transputer targets.
`occonf`	The configurer which creates configuration binary files from configuration descriptions.

Default file extensions

Extension	Description
.btl	Bootable code file. Created by `icollect`.
.btr	Executable code minus bootstrap information used for input to `ieprom`. Created by `icollect`.
.cfb	Configuration binary file. Created by `occonf` or `icollect`.
.clu	Configuration object file. Created by `occonf`.
.dmp	Core-dump file created by `idump` or network-dump file created by `idebug` or `isim`.
.inc	Include file. Input to `oc` and `occonf`.
.lku	Linked unit. Created by `ilink`.
.lbb	Library build file. Command file for `ilibr`.
.lib	Library file. Created by `ilibr`.
.liu	Library usage file. Created by `imakef`.
.lnk	Linker indirect file. Command file for `ilink`.
.occ	occam 2 source files.
.pgm	Configuration description source file.
.rsc	Dynamically loadable code file. Created by `icollect`.
.tco	Compiled code file. Created by `oc`.

March 1991

Tools

`icollect` – code collector

Generates bootable code files for single and multiple transputer programs from linked units and configuration binary files respectively. Also used to create non-bootable single transputer programs for dynamic loading or booting from ROM.

Syntax: `icollect` *filename* {*options*}

where: *filename* is a linked unit (`.lku`) or configuration binary file (`.cfb`).

Note: If the input file is a linked unit (single transputer mode) then the '`T`' option must be specified.

Options:

`B` *filename*	Specifies external bootstrap loader program.
`C` *filename*	Specifies debug data file.
`D`	Disables the debug data file for single transputer programs.
`E`	Changes setting of Halt On Error flag.
`I`	Displays progress information.
`K`	Creates non-bootstrapped code.
`L`	Loads the tool and terminates.
`M` *memorysize*	Specifies memory size on root processor.
`O` *filename*	Specifies output file.
`P` *filename*	Specifies a memory map output file.
`RA`	Creates RAM-loadable code.
`RO`	Creates ROM-loadable code.
`RS` *romsize*	Specifies size of ROM on the root processor.
`S` *stacksize*	Specifies extra runtime stack size for single transputer programs.

Options:

T	Creates bootable code for a single transputer.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.
Y	Disables interactive debugging with `idebug` for single transputer programs.

`icvemit` – memory interface convertor

Converts memory configuration files produced by `iemi` (a previous version of `iemit`) to a file format suitable for use with `iemit` and `ieprom`.

Syntax: `icvemit` *filename* {*options*}

where: *filename* is the input file to be converted.

Options:

`I`	Displays progress information.
`o` *filename*	Specifies output file.

`icvlink` – object code convertor

Converts object code into TCOFF format.

Syntax: `icvlink` *filename* {*options*}

where: *filename* is the object file to be converted.

Options:

D	Creates multiple modules from TA and TC modules. For libraries only.
I	Displays progress information.
L	Loads the tool and terminates.
O *filename*	Specifies output file.
P	Creates T8 modules from TA and TC modules.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.

March 1991

`idebug`– network debugger

Provides post-mortem debugging and breakpoint debugging.

Syntax: `idebug` *filename* {*options*}

where: *filename* is a bootable file.

Options:

B *linknumber*	Breakpoint debug a network.
M *linknumber*	Postmortem debug a previous breakpoint session.
T *linknumber*	Postmortem debug a program not using the root processor.
R *filename*	Postmortem debug a program using the root transputer.
N *filename*	Postmortem debug from a network dump file.
C *type*	Specify processor type instead of class for non-configured programs.
D	Dummy debugging session.
A	Assert subsystem analyse on the network.
S	Ignore subsystem error status when breakpoint debugging.
I	Display debugger version number.

idump – memory dumper

Writes the root transputer's memory to a file. Used in debugging programs that use the root transputer.

Syntax: idump *filename memsize {offset length}*

where: *filename* is the bootable program file

memsize is the amount of memory measured in bytes to be dumped to the file

offset is a byte offset from the start of memory

length is the number of bytes of memory starting at *offset* to be written to the file in addition to *memsize*.

`iemit` – memory configurer

Evaluates memory configurations.

Syntax: `iemit` *{options}*

Options:

A	Produce ASCII output file.
E	Invoke interactive mode.
F *filename*	Specify input memory configuration file.
I	Displays progress information.
O *filename*	Specify output filename.
P	Produce PostScript output file.

`ieprom`– EPROM program convertor

Formats bootable code for installation by ROM loaders.

Syntax: `ieprom` *filename* { *option* }

where: *filename* is the control file.

Options:

`I`	Displays progress information.

`ilibr` – librarian

Builds libraries of code from separate files.

Syntax: `ilibr`   {*filenames*} {*options*}

where: *filenames* is a list of compiled modules.

Options:

`F` *filename*	Specifies library indirect file.
`I`	Displays progress information.
`L`	Loads the tool and terminates.
`O` *filename*	Specifies output file.
`XM`	Loads transputer-hosted tool for multiple execution.
`XO`	Loads transputer-hosted tool for single execution.

`ilink` – linker

Links object files together, resolving external references, to create a single linked unit.

Syntax: `ilink`   {*filenames*}   {*options*}

where: *filenames* is a list of compiled modules or libraries.

Options:

TA	Link for class TA (T400/T414/T425/T800/T801/T805).
TB	Link for class TB (T400/T414/T425).
T212	Link for T212.
T2	Link for T212/T222/M212.
T222	Link for T222.
T225	Link for T225.
T3	Link for T225.
T400	Link for T400.
T414	Link for T414. Default.
T4	Link for T414.
T425	Link for T425.
T5	Link for T425/T400.
T800	Link for T800.
T8	Link for T800.
T801	Link for T801.
T805	Link for T805.
T9	Link for T801/T805.
H	Generates HALT mode output. Default.
S	Generates STOP mode output.
X	Generates UNIVERSAL mode output.

Options:

`T`	Specifies TCOFF format. Default.
`LB`	Specifies LFF format. For use with the `iboot` and iconf tools, supported by the D705, D605 and D505 toolsets.
`LC`	Specifies LFF format. As above, but only for use with the `iconf` tool.
`EX`	Allows extraction of modules without linking them.
`F` *filename*	Specifies linker indirect file.
`I`	Displays progress information.
`KB` *memorysize*	Specifies maximum internal code buffer size.
`L`	Loads the tool and terminates.
`ME` *entryname*	Specifies main entry point.
`MO` *filename*	Generates module information file.
`O` *filename*	Specifies output file.
`U`	Allows unresolved references.
`XM`	Loads transputer-hosted tool for multiple execution.
`XO`	Loads transputer-hosted tool for single execution.
`Y`	Disables interactive debugging for occam code.

ilist – binary lister

Decodes and displays information from object files and bootable files.

Syntax: ilist *filename* {*options*}

where: *filename* is an object or bootable file.

Options:

A	Displays all available symbolic information.
C	Displays code in hexadecimal.
E	Displays all exported names.
H	Displays specified file(s) in hexadecimal format.
I	Displays full progress information.
L	Loads the tool and terminates.
M	Displays module data.
N	Displays library index data.
O *filename*	Specifies output file.
P	Displays procedural interfaces.
R *reference*	Displays library module(s) with the specified reference.
T	Displays full listing.
W	Identifies file type. Default.
X	Displays all external references.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.

`imakef` – Makefile generator

Creates Makefiles for toolset compilations.

Syntax: `imakef` {*filenames*} {*options*}

where: *filenames* is a list of target object or bootable files.

Options:

C	For C modules only. Specifies input from a linker indirect file.
D	Disables debugging data.
I	Displays full progress information.
L	Loads the tool and terminates.
NI	Do not include dependencies on `ISEARCH`.
O *filename*	Specifies output file.
R	Incorporates a delete rule.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.
Y	Disables interactive (breakpoint) debugging in the target file.

`iserver` – server/loader

Loads programs onto transputers and transputer boards and serves host communications.

Syntax: `iserver` *bootablefile* {*options*}

where: *bootablefile* is a bootable file.

Options:

`SA`	Analyses root transputer and peeks 8K of memory.
`SB` *filename*	Loads specified program.
`SC` *filename*	Copies specified file to root transputer link.
`SE`	Monitors transputer error flag.
`SI`	Displays progress information.
`SL` *name*	Specifies device name or link address.
`SP` *n*	Specifies KBytes of memory peeked on Analyse.
`SR`	Resets root transputer and subsystem on the link.
`SS`	Serves the link, that is, starts up the host communications.
Option `SB` is equivalent to invoking the combination: `SR SS SI SC` *filename*.	

`isim`– T425 simulator

Simulates the execution of a program on the IMS T425.

Syntax: `isim` *filename programparameters {options}*

where: *filename* is a bootable file.

Options:

B	Batch mode operation.
BQ	Batch Quiet mode. No progress information.
BV	Batch Verify mode.
I	Displays full progress information.
L	Loads the tool and terminates.
N	No more options for the simulator.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.

`iskip`– skip loader

Allows programs to be loaded onto transputer networks beyond the root transputer.

Syntax: `iskip` *linknumber* {*options*}

where: *linknumber* is the root transputer link to which the target network is connected.

Options:

`E`	Monitors subsystem error status; terminates when it becomes set.
`R`	Reset subsystem but not the root transputer.
`I`	Displays detailed progress information.

oc – occam 2 compiler

Compiles occam 2 source code.

Syntax: oc   *filename*   {*options*}

where: *filename* is an occam source file.

Options:

TA	Compile for class TA (T400/T414/T425/T800/T801/T805).
TB	Compile for class TB (T400/T414/T425).
T212	Compile for T212.
T2	Compile for T212/T222/M212.
T222	Compile for T222.
T225	Compile for T225.
T3	Compile for T225.
T400	Compile for T400.
T414	Compile for T414. Default.
T4	Compile for T414.
T425	Compile for T425.
T5	Compile for T425/T400.
T800	Compile for T800.
T8	Compile for T800.
T801	Compile for T801.
T805	Compile for T805.
T9	Compile for T801/T805.
H	Generates HALT mode output. Default.
S	Generates STOP mode output.
X	Generates UNIVERSAL mode output.
G	Enables use of ASM or GUY code for a restricted set of transputer instructions.
W	Enables use of ASM or GUY code for the full set of transputer instructions.

Options:

K	Disables insertion of run-time range checks.
U	Disables insertion of run-time error checks.
NA	Disables insertion of run-time checks for **ASSERT**s.
L	Loads the tool and terminates.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.
A	Disables alias and usage checking.
B	Displays messages in brief.
C	Only performs check.
D	Generates minimal debugging information.
E	Disables use of the compiler libraries.
I	Displays progress information.
N	Disables usage checking.
O *outputfile*	Specifies output file.
R *filename*	Redirects error messages to a file.
V	Disables the generation of code with a separate vector space requirement.
Y	Disables interactive debugging with **idebug**.
NWP	Disables warning messages of unused parameters.
NWU	Disables warning messages of unused variables or routines.
WD	Provides a warning when a name is descoped.
WO	Provides a warning when a run-time alias check is generated.

`occonf` – configurer

Generates configuration binary files frrom configuration descriptions.

Syntax: `occonf` *filename* {*options*}

where: *filename* is a configuration description (`.pgm`) source file.

Options:

`B`	Displays messages in brief.
`C`	Only performs check.
`I`	Displays progress information.
`L`	Loads the tool and terminates.
`O` *outputfile*	Specifies output file.
`R` *filename*	Redirects error messages to a file.
`V`	Disables the generation of code with a separate vector space requirement.
`Y`	Disables interactive debugging with `idebug`.
`RA`	Creates RAM-loadable code.
`RO`	Creates ROM-loadable code.
`H`	Generates HALT mode output. Default.
`S`	Generates STOP mode output.
`X`	Generates UNIVERSAL mode output.
`K`	Disables insertion of run-time range checks.
`U`	Disables insertion of run-time error checks.
`NA`	Disables insertion of run-time checks for `ASSERTs`.

Options:

|	
G	Enables use of ASM or GUY code for a restricted set of transputer instructions.
W	Enables use of ASM or GUY code for the full set of transputer instructions.
RE	Enables memory lay-out re-ordering.
NWP	Disables warning messages of unused parameters.
NWU	Disables warning messages of unused variables or routines.
WD	Provides a warning when a name is descoped.
WO	Provides a warning when a run-time alias is check is generated.
XM	Loads transputer-hosted tool for multiple execution.
XO	Loads transputer-hosted tool for single execution.

Debugger commands

Debugger symbolic functions

Function	Description
INSPECT	Inspect symbol.
CHANNEL	Locate to process waiting on channel.
TOP	Locate to last error or location.
RETRACE	Retrace last operation.
RELOCATE	Locate back to last location line.
INFO	Display extra information.
SEARCH	Search for string.
MONITOR	Change to Monitor page.
BACKTRACE	Locate to procedure or function call.
HELP	Display function keys.
GET ADDRESS	Display address of source line.
CHANGE FILE	Display different file.
ENTER FILE	Display included file.
EXIT FILE	Display enclosing file.
GOTO LINE	Go to specific line in file.
TOP OF FILE	Go to first line in file.
BOTTOM OF FILE	Go to last line in file.
TOGGLE HEX	Display C variables in Hex.
TOGGLE BREAK ‡	Set/clear breakpoint.
MODIFY ‡	Change variable in memory.
RESUME ‡	Resume program from breakpoint.
CONTINUE FROM ‡	Resume program from current line.
INTERRUPT ‡	Return to Monitor page.
MODIFY ‡	Change variable in memory.
FINISH	Quit.
‡ Breakpoint mode only	

Debugger Monitor page commands

Key	Meaning	Description
A	ASCII	View memory in ASCII.
B‡	Breakpoints	Enter breakpoint debugging.
C	Compare	Compare actual code with expected code.
D	Disassemble	Display transputer instructions.
E	Next Error	Go to next processor with error flag set.
F	Select file	Select source file for display.
G	Goto process	Enter source level debugging for a process.
H	Hex	View memory in hexadecimal.
I	Inspect	View memory in any type.
J‡	Jump	Start or resume program.
K	Processor names	Display processor names.
L	Links	Display processes waiting on links or **Event** pin.
M	Memory map	Display transputer's memory map.
N	Network dump	Dump network memory.
O	Specify process	Resume symbolic debugging.
P	Processor	Change processor.
Q	Quit	Quit debugger.
R	Run queues	Display active process queues.
T	Timer queues	Display timer queues.
U‡	Update	Update Monitor page with new register values.
V	Process names	Display process names.
W‡	Write	Write to memory.
X	Exit	Change to symbolic mode.
Y‡	Postmortem	Change to post-mortem debugging.
?	Help	Display help information.
‡ Breakpoint mode only		

Debugger Monitor page commands (contd)

Function	Description
RETRACE	Redo last operation.
RELOCATE	Locate to last location line.
LINE UP	Next line.
LINE DOWN	Previous line.
PAGE UP	Previous page.
PAGE DOWN	Next page.
HELP	Display help information.
REFRESH	Re-draw the screen.
TOP	Find last instruction executed.
↑	Scroll display.
↓	Scroll display.
←	Scroll processor left.
→	Scroll processor right.

Simulator commands

Key	Meaning	Description
A	ASCII	View memory in ASCII.
B	Breakpoints	Breakpoint menu.
D	Disassemble	Display transputer instructions.
G	Go	Run/resume program.
H	Hex	View memory in hexadecimal.
I	Inspect	Display memory in occam type.
J	Jump	Resume (Jump into) program.
L	Links	Display processes waiting on links or **Event** pin.
M	Memory map	Display transputer memory map.
N	Network dump	Create core dump file.
P	Program boot	Simulate a program load.
Q	Quit	Quit simulator.
R	Run queues	Display active process queues.
S	Single step	Single step one transputer instruction.
T	Timer queues	Display timer queues.
U	Assign register	Assign value to register.
?	Help	Display help information.
? ♯	Query	Display registers and queue pointers.
. ♯	Where	Display next `Iptr`.
⊡↑		Scroll display.
⊡↓		Scroll display.
HELP		Display help information.
REFRESH		Redraw the screen.
FINISH		Quit simulator.
♯ Batch mode commands.		

Libraries

User libraries

Library	Description
hostio.lib	Host file server library.
streamio.lib	Stream i/o library.
snglmath.lib	Single length maths library.
dblmath.lib	Double length maths library.
tbmaths.lib	T400/T414/T425 optimised maths functions.
string.lib	String handling library.
convert.lib	Type conversion library.
xlink.lib	Extraordinary link handling library.
crc.lib	Block CRC library.
debug.lib	Debugging support library.
callc.lib	Mixed languages support library.
msdos.lib	DOS specific hostio library.

March 1991

Include files

File	Contents
`hostio.inc`	Host file server constants.
`streamio.inc`	Stream i/o constants.
`mathvals.inc`	Maths and trigonometric constants.
`linkaddr.inc`	Transputer link addresses.
`ticks.inc`	Rates of the two transputer clocks.
`msdos.inc`	DOS specific hostio library constants.

Compiler libraries

File	Processor type
`occam2.lib`	T212/T222/T225/M212
`occama.lib`	T400/T414/T425/TA/TB
`occam8.lib`	T800/T801/T805
`occamutl.lib`	All.
`virtual.lib`	All.

The compiler libraries support all error modes. `occamutl.lib` contains routines called by some of the other compiler libraries. `virtual.lib` supports interactive debugging.

Bit manipulation functions

Result	Function	Parameter Specifiers
INT	BITCOUNT	VAL INT Word, CountIn
INT	BITREVNBITS	VAL INT x, n
INT	BITREVWORD	VAL INT x

2D block moves

Procedure	Parameter Specifiers
CLIP2D	VAL [][]BYTE Source, VAL INT sx, sy, [][]BYTE Dest, VAL INT dx, dy, width, length
DRAW2D	VAL [][]BYTE Source, VAL INT sx, sy, [][]BYTE Dest, VAL INT dx, dy, width, length
MOVE2D	VAL [][]BYTE Source, VAL INT sx, sy, [][]BYTE Dest, VAL INT dx, dy, width, length

CRC functions

Result	Function	Parameter Specifiers
INT	CRCWORD	VAL INT data, CRCIn, generator
INT	CRCBYTE	VAL INT data, CRCIn, generator

Supplementary arithmetic support functions

Result(s)	Function	Parameter specifiers
INT	FRACMUL	VAL INT x, y
INT, INT, INT	UNPACKSN	VAL INT x
INT	ROUNDSN	VAL INT Yexp, Yfrac, Yguard

Hostio library   #USE "hostio.lib"

Procedure	Parameter Specifiers
so.ask	CHAN OF SP fs, ts, VAL []BYTE prompt, replies, VAL BOOL display.possible.replies, VAL BOOL echo.reply, INT reply.number
so.buffer	CHAN OF SP fs, ts, from.user, to.user, CHAN OF BOOL stopper
so.close	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result
so.commandline	CHAN OF SP fs, ts, VAL BYTE all, INT length, []BYTE string, BYTE result
so.core	CHAN OF SP fs, ts, VAL INT32 offset, INT bytes.read, []BYTE data, BYTE result
so.date.to.ascii	VAL [so.date.len]INT date, VAL BOOL long.years, VAL BOOL days.first, [so.time.string.len]BYTE string
so.eof	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result
so.exit	CHAN OF SP fs, ts, VAL INT32 status
so.ferror	CHAN OF SP fs, ts, VAL INT32 streamid, INT32 error.no, INT length, []BYTE message, BYTE result
so.flush	CHAN OF SP fs, ts, VAL INT32 streamid, BYTE result

Procedure	Parameter Specifiers

`so.fwrite.char`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL BYTE char, BYTE result`

`so.fwrite.hex.int`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL INT n, width, BYTE result`

`so.fwrite.hex.int32`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid, n,`
	`VAL INT width, BYTE result`

`so.fwrite.hex.int64`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid, VAL INT64 n,`
	`VAL INT width, BYTE result`

`so.fwrite.int`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL INT n, field, BYTE result`

`so.fwrite.int32`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid, VAL INT32 n,`
	`VAL INT field, BYTE result`

`so.fwrite.int64`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid, VAL INT64 n,`
	`VAL INT field, BYTE result`

`so.fwrite.nl`	`CHAN OF SP fs, ts,` `VAL INT32 streamid, BYTE result`

`so.fwrite.real32`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL REAL32 r, VAL INT Ip, Dp,`
	`BYTE result`

`so.fwrite.real64`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL REAL64 r, VAL INT Ip, Dp,`
	`BYTE result`

`so.fwrite.string`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL []BYTE string, BYTE result`

`so.fwrite.string.nl`	`CHAN OF SP fs, ts,`
	`VAL INT32 streamid,`
	`VAL []BYTE string, BYTE result`

Procedure	Parameter Specifiers
so.getenv	CHAN OF SP fs, ts, VAL []BYTE name, INT length, []BYTE value, BYTE result
so.getkey	CHAN OF SP fs, ts, BYTE key, result
so.gets	CHAN OF SP fs, ts, VAL INT32 streamid, INT length, []BYTE data, BYTE result
so.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper
so.open	CHAN OF SP fs, ts, VAL []BYTE name, VAL BYTE type, mode, INT32 streamid, BYTE result
so.open.temp	CHAN OF SP fs, ts, VAL BYTE type, [so.temp.filename.length] BYTE filename, INT32 streamid, BYTE result
so.overlapped.buffer	CHAN OF SP fs, ts, from.user, to.user, CHAN OF BOOL stopper
so.overlapped.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper, []INT queue
so.overlapped.pri.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper, []INT queue

Procedure	Parameter Specifiers
so.parse.command.line	CHAN OF SP fs, ts, VAL [][]BYTE option.strings, VAL []INT option.parameters.required, []BOOL option.exists, [][2]INT option.parameters, INT error.len, []BYTE line
so.pollkey	CHAN OF SP fs, ts, BYTE key, result
so.popen.read	CHAN OF SP fs, ts, VAL []BYTE filename, VAL []BYTE path.variable.name, VAL BYTE open.type, INT full.len, []BYTE full.name, INT32 streamid, BYTE result
so.pri.multiplexor	CHAN OF SP fs, ts, []CHAN OF SP from.user, to.user, CHAN OF BOOL stopper
so.puts	CHAN OF SP fs, ts, VAL INT32 streamid, VAL []BYTE data, BYTE result
so.read	CHAN OF SP fs, ts, VAL INT32 streamid, INT length, []BYTE data
so.read.echo.any.int	CHAN OF SP fs, ts, INT n, BOOL error
so.read.echo.hex.int	CHAN OF SP fs, ts, INT n, BOOL error
so.read.echo.hex.int32	CHAN OF SP fs, ts, INT32 n, BOOL error
so.read.echo.hex.int64	CHAN OF SP fs, ts, INT64 n, BOOL error
so.read.echo.int	CHAN OF SP fs, ts, INT n, BOOL error
so.read.echo.int32	CHAN OF SP fs, ts, INT32 n, BOOL error

Procedure	Parameter Specifiers
so.read.echo.int64	CHAN OF SP fs, ts, INT64 n, BOOL error
so.read.echo.line	CHAN OF SP fs, ts, INT len, []BYTE line, BYTE result
so.read.echo.real32	CHAN OF SP fs, ts, REAL32 n, BOOL error
so.read.echo.real64	CHAN OF SP fs, ts, REAL64 n, BOOL error
so.read.line	CHAN OF SP fs, ts, INT len, []BYTE line, BYTE result
so.remove	CHAN OF SP fs, ts, VAL []BYTE name, BYTE result
so.rename	CHAN OF SP fs, ts, VAL []BYTE oldname, newname, BYTE result
so.seek	CHAN OF SP fs, ts, VAL INT32 streamid, VAL INT32 offset, origin, BYTE result
so.system	CHAN OF SP fs, ts, VAL []BYTE command, INT32 status, BYTE result
so.tell	CHAN OF SP fs, ts, VAL INT32 streamid, INT32 position, BYTE result
so.test.exists	CHAN OF SP fs, ts, VAL []BYTE filename, BOOL exists
so.time	CHAN OF SP fs, ts, INT32 localtime, UTCtime
so.time.to.ascii	VAL INT32 time, VAL BOOL long.years, VAL BOOL days.first [so.time.string.len]BYTE string
so.time.to.date	VAL INT32 input.time, [so.date.len]INT date

Procedure	Parameter Specifiers
so.today.ascii	CHAN OF SP fs, ts, VAL BOOL long.years, days.first, [so.time.string.len]BYTE string
so.today.date	CHAN OF SP fs, ts, [so.date.len]INT date
so.version	CHAN OF SP fs, BYTE version, host, os, board
so.write	CHAN OF SP fs, ts, VAL INT32 streamid, VAL []BYTE data, INT length
so.write.char	CHAN OF SP fs, ts, VAL BYTE char
so.write.hex.int	CHAN OF SP fs, ts, VAL INT n, width
so.write.hex.int32	CHAN OF SP fs, ts, VAL INT32 n, VAL INT width
so.write.hex.int64	CHAN OF SP fs, ts, VAL INT64 n, VAL INT width
so.write.int	CHAN OF SP fs, ts, VAL INT n, field
so.write.int32	CHAN OF SP fs, ts, VAL INT32 n, VAL INT field
so.write.int64	CHAN OF SP fs, ts, VAL INT64 n, VAL INT field
so.write.nl	CHAN OF SP fs, ts
so.write.real32	CHAN OF SP fs, ts, VAL REAL32 r, VAL INT Ip, Dp
so.write.real64	CHAN OF SP fs, ts, VAL REAL64 r, VAL INT Ip, Dp
so.write.string	CHAN OF SP fs, ts, VAL []BYTE string
so.write.string.nl	CHAN OF SP fs, ts, VAL []BYTE string

Streamio library

#USE "streamio.lib"

Procedure	Parameter Specifiers
ks.keystream.sink	CHAN OF KS keys
ks.keystream.to.scrstream	CHAN OF KS keyboard, CHAN OF SS scrn
ks.read.char	CHAN OF KS source, INT char
ks.read.int	CHAN OF KS source, INT number, char
ks.read.int64	CHAN OF KS source, INT64 number, INT char
ks.read.line	CHAN OF KS source, INT len, []BYTE line, INT char
ks.read.real32	CHAN OF KS source, REAL32 number, INT char
ks.read.real64	CHAN OF KS source, REAL64 number, INT char
so.keystream.from.file	CHAN OF SP fs, ts, CHAN OF KS keys.out, VAL []BYTE filename, BYTE result
so.keystream.from.kbd	CHAN OF SP fs, ts, CHAN OF KS keys.out, CHAN OF BOOL stopper, VAL INT ticks.per.poll
so.keystream.from.stdin	CHAN OF SP fs, ts, CHAN OF KS keys.out, BYTE result
so.scrstream.to.ANSI	CHAN OF SP fs, ts, CHAN OF SS scrn

Procedure	Parameter Specifiers
so.scrstream.to.file	CHAN OF SP fs, ts, CHAN OF SS scrn, VAL []BYTE filename, BYTE result
so.scrstream.to.stdout	CHAN OF SP fs, ts, CHAN OF SS scrn, BYTE result
so.scrstream.to.TVI920	CHAN OF SP fs, ts, CHAN OF SS scrn
ss.beep	CHAN OF SS scrn
ss.clear.eol	CHAN OF SS scrn
ss.clear.eos	CHAN OF SS scrn
ss.del.line	CHAN OF SS scrn
ss.delete.chr	CHAN OF SS scrn
ss.delete.chl	CHAN OF SS scrn
ss.down	CHAN OF SS scrn
ss.goto.xy	CHAN OF SS scrn, VAL INT x, y
ss.ins.line	CHAN OF SS scrn
ss.insert.char	CHAN OF SS scrn, VAL BYTE ch
ss.left	CHAN OF SS scrn
ss.right	CHAN OF SS scrn
ss.scrstream.copy	CHAN OF SS scrn.in, scrn.out
ss.scrstream.fan.out	CHAN OF SS scrn, screen.out1, screen.out2
ss.scrstream.from.array	CHAN OF SS scrn, VAL []BYTE buffer
ss.scrstream.multiplexor	[]CHAN OF SS screen.in, CHAN OF SS screen.out, CHAN OF INT stopper
ss.scrstream.sink	CHAN OF SS scrn

Procedure	Parameter Specifiers
ss.scrstream.to.array	CHAN OF SS scrn, []BYTE buffer
ss.up	CHAN OF SS scrn
ss.write.char	CHAN OF SS scrn, VAL BYTE char
ss.write.endstream	CHAN OF SS scrn
ss.write.hex.int	CHAN OF SS scrn, VAL INT number, field
ss.write.hex.int64	CHAN OF SS scrn, VAL INT64 number, VAL INT field
ss.write.int	CHAN OF SS scrn, VAL INT number, field
ss.write.int64	CHAN OF SS scrn, VAL INT64 number, VAL INT field
ss.write.nl	CHAN OF SS scrn
ss.write.string	CHAN OF SS scrn, VAL []BYTE str
ss.write.real32	CHAN OF SS scrn, VAL REAL32 number, VAL INT Ip, Dp
ss.write.real64	CHAN OF SS scrn, VAL REAL64 number, VAL INT Ip, Dp
ss.write.text.line	CHAN OF SS scrn, VAL []BYTE str

Single length maths library   #USE "snglmath.lib"

Result(s)	Function	Parameter specifiers
REAL32	ACOS	VAL REAL32 X
REAL32	ALOG	VAL REAL32 X
REAL32	ALOG10	VAL REAL32 X
REAL32	ASIN	VAL REAL32 X
REAL32	ATAN	VAL REAL32 X
REAL32	ATAN2	VAL REAL32 X, VAL REAL32 Y
REAL32	COS	VAL REAL32 X
REAL32	COSH	VAL REAL32 X
REAL32	EXP	VAL REAL32 X
REAL32	POWER	VAL REAL32 X, VAL REAL32 Y
REAL32, INT32	RAN	VAL INT32 X
REAL32	SIN	VAL REAL32 X
REAL32	SINH	VAL REAL32 X
REAL32	TAN	VAL REAL32 X
REAL32	TANH	VAL REAL32 X

Double length maths library #USE "dblmath.lib"

Result(s)	Function	Parameter specifiers
REAL64	DACOS	VAL REAL64 X
REAL64	DALOG	VAL REAL64 X
REAL64	DALOG10	VAL REAL64 X
REAL64	DASIN	VAL REAL64 X
REAL64	DATAN	VAL REAL64 X
REAL64	DATAN2	VAL REAL64 X, VAL REAL64 Y
REAL64	DCOS	VAL REAL64 X
REAL64	DCOSH	VAL REAL64 X
REAL64	DEXP	VAL REAL64 X
REAL64	DPOWER	VAL REAL64 X, VAL REAL64 Y
REAL64,INT64	DRAN	VAL INT64 X
REAL64	DSIN	VAL REAL64 X
REAL64	DSINH	VAL REAL64 X
REAL64	DTAN	VAL REAL64 X
REAL64	DTANH	VAL REAL64 X

T400/T414/T425 maths library #USE "tbmaths.lib"

Contains the same functions as snglmath.lib and dblmath.lib, but optimised for the IMS T400, IMS T414 and IMS T425 procesoors.

String library   #USE "string.lib"

Result	Function	Parameter Specifiers
INT	char.pos	VAL BYTE search, VAL []BYTE str
INT	compare.strings	VAL []BYTE str1, str2
BOOL	eqstr	VAL []BYTE s1,s2
BOOL	is.digit	VAL BYTE char
BOOL	is.hex.digit	VAL BYTE char
BOOL	is.id.char	VAL BYTE char
BOOL	is.in.range	VAL BYTE char, bottom, top
BOOL	is.lower	VAL BYTE char
BOOL	is.upper	VAL BYTE char
INT, BYTE	search.match	VAL []BYTE possibles, str
INT, BYTE	search.no.match	VAL []BYTE possibles, str
INT	string.pos	VAL []BYTE search, str

Procedure	Parameter Specifiers
append.char	INT len, []BYTE str, VAL BYTE char
append.hex.int	INT len, []BYTE str, VAL INT number, field
append.hex.int64	INT len, []BYTE str, VAL INT64 number, VAL INT width
append.int	INT len, []BYTE str, VAL INT number, field
append.int64	INT len, []BYTE str, VAL INT64 number, VAL INT field
append.real32	INT len, []BYTE str, VAL REAL32 number, VAL INT Ip, Dp
append.real64	INT len, []BYTE str, VAL REAL64 number, VAL INT Ip, Dp
append.text	INT len, []BYTE str, VAL []BYTE text
delete.string	INT len, []BYTE str, VAL INT start, size, BOOL not.done
insert.string	VAL []BYTE new.str, INT len, []BYTE str, VAL INT start, BOOL not.done
next.int.from.line	VAL []BYTE line, INT ptr, number, BOOL ok
next.word.from.line	VAL []BYTE line, INT ptr, INT len, []BYTE word, BOOL ok
str.shift	[]BYTE str, VAL INT start, len, shift, BOOL not.done
to.lower.case	[]BYTE str
to.upper.case	[]BYTE str

Type conversion library   #USE "convert.lib"

Procedure	Parameter Specifiers
BOOLTOSTRING	INT len, []BYTE string, VAL BOOL b
HEXTOSTRING	INT len, []BYTE string, VAL INT n
HEX16TOSTRING	INT len, []BYTE string, VAL INT16 n
HEX32TOSTRING	INT len, []BYTE string, VAL INT32 n
HEX64TOSTRING	INT len, []BYTE string, VAL INT64 n
INTTOSTRING	INT len, []BYTE string, VAL INT n
INT16TOSTRING	INT len, []BYTE string, VAL INT16 n
INT32TOSTRING	INT len, []BYTE string, VAL INT32 n
INT64TOSTRING	INT len, []BYTE string, VAL INT64 n
REAL32TOSTRING	INT len, []BYTE string, VAL REAL32 X, VAL INT Ip, Dp
REAL64TOSTRING	INT len, []BYTE string, VAL REAL64 X, VAL INT Ip, Dp
STRINGTOBOOL	BOOL Error, b, VAL []BYTE string
STRINGTOHEX	BOOL Error, INT n, VAL []BYTE string
STRINGTOHEX16	BOOL Error, INT16 n, VAL []BYTE string
STRINGTOHEX32	BOOL Error, INT32 n, VAL []BYTE string
STRINGTOHEX64	BOOL Error, INT64 n, VAL []BYTE string
STRINGTOINT	BOOL Error, INT n, VAL []BYTE string
STRINGTOINT16	BOOL Error, INT16 n, VAL []BYTE string
STRINGTOINT32	BOOL Error, INT32 n, VAL []BYTE string
STRINGTOINT64	BOOL Error, INT64 n, VAL []BYTE string
STRINGTOREAL32	BOOL Error, REAL32 X, VAL []BYTE string
STRINGTOREAL64	BOOL Error, REAL64 X, VAL []BYTE string

Block CRC library   #USE "crc.lib"

Result	Function	Parameter Specifiers
INT	CRCFROMLSB	VAL []BYTE InputString VAL INT PolynomialGenerator, VAL INT OldCRC
INT	CRCFROMMSB	VAL []BYTE InputString, VAL INT PolynomialGenerator, VAL INT OldCRC

Link handling library   #USE "xlink.lib"

Procedure	Parameter Specifiers
InputOrFail.c	CHAN OF ANY c, []BYTE mess CHAN OF INT kill, BOOL aborted
InputOrFail.t	CHAN OF ANY c, []BYTE mess, TIMER t, VAL INT time, BOOL aborted
OutputOrFail.c	CHAN OF ANY c, VAL []BYTE mess, CHAN OF INT kill, BOOL aborted
OutputOrFail.t	CHAN OF ANY c, VAL []BYTE mess, TIMER t, VAL INT time, BOOL aborted
Reinitialise	CHAN OF ANY c

Debugging support library   #USE "debug.lib"

Procedure	Parameter Specifiers
DEBUG.ASSERT	VAL BOOL assertion
DEBUG.MESSAGE	VAL []BYTE message
DEBUG.STOP	–
DEBUG.TIMER	CHAN OF INT stop

Mixed languages support library #USE "callc.lib"

Procedure	Parameter Specifiers
init.static	[] INT static.area, INT required.size, gsb
init.heap	VAL INT gsb, []INT heap.area
terminate.heap.use	VAL INT gsb
terminate.static.use	VAL INT gsb

DOS specific hostio library   #USE "msdos.lib"

Procedure	Parameter Specifiers
dos.receive.block	CHAN OF SP fs, ts, VAL INT32 location, INT bytes.read, []BYTE block, BYTE result
dos.send.block	CHAN OF SP fs, ts, VAL INT32 location, VAL []BYTE block, INT len, BYTE result
dos.call.interrupt	CHAN OF SP fs, ts, VAL INT16 interrupt, VAL [dos.interrupt.regs.size] BYTE register.block.in, BYTE carry.flag, [dos.interrupt.regs.size] BYTE register.block.out, BYTE result
dos.read.regs	CHAN OF SP fs, ts, [dos.read.regs.size] BYTE registers, BYTE result
dos.port.read	CHAN OF SP fs, ts, VAL INT16 port.location, BYTE value, result
dos.port.write	CHAN OF SP fs, ts, VAL INT16 port.location, VAL BYTE value, BYTE result

Compiler library user functions

Function	Result(s)	Parameter specifiers
ABS	REAL32	VAL REAL32 x
ARGUMENT.REDUCE	BOOL, INT32, REAL32	VAL REAL32 x, y, y.err
ASHIFTRIGHT	INT	VAL INT argument, places
ASHIFTLEFT	INT	VAL INT argument, places
COPYSIGN	REAL32	VAL REAL32 x, y
DABS	REAL64	VAL REAL64 x
DARGUMENT.REDUCE	BOOL, INT32, REAL64	VAL REAL64 x, y, y.err
DCOPYSIGN	REAL64	VAL REAL64 x, y
DDIVBY2	REAL64	VAL REAL64 x
DFLOATING.UNPACK	INT, REAL64	VAL REAL64 x
DFPINT	REAL64	VAL REAL64 x
DIEEECOMPARE	INT	VAL REAL64 x, y
DISNAN	BOOL	VAL REAL64 x
DIVBY2	REAL32	VAL REAL32 x
DLOGB	REAL64	VAL REAL64 x
DMINUSX	REAL64	VAL REAL64 x
DMULBY2	REAL64	VAL REAL64 x
DNEXTAFTER	REAL64	VAL REAL64 x, y
DNOTFINITE	BOOL	VAL REAL64 x
DORDERED	BOOL	VAL REAL64 x, y
DSCALEB	REAL64	VAL REAL64 x, VAL INT n
DSQRT	REAL64	VAL REAL64 x
FLOATING.UNPACK	INT, REAL32	VAL REAL32 x
FPINT	REAL32	VAL REAL32 x

Function	Result(s)	Parameter specifiers
IEEE32OP	BOOL, REAL32	VAL REAL32 x, VAL INT rm, op, VAL REAL32 y
IEEE32REM	BOOL, REAL32	VAL REAL32 X, Y
IEEE64OP	BOOL, REAL64	VAL REAL64 x, VAL INT rm, op, VAL REAL64 y
IEEE64REM	BOOL, REAL64	VAL REAL64 X, Y
IEEECOMPARE	INT	VAL REAL32 x, y
ISNAN	BOOL	VAL REAL32 x
LOGB	REAL32	VAL REAL32 x
LONGADD	INT	VAL INT left, right, carry.in
LONGDIFF	INT, INT	VAL INT left, right, borrow.in
LONGDIV	INT, INT	VAL INT dividend.hi, dividend.lo, divisor
LONGPROD	INT, INT	VAL INT left, right, carry.in
LONGSUB	INT	VAL INT left, right, borrow.in
LONGSUM	INT, INT	VAL INT left, right, carry.in
MINUSX	REAL32	VAL REAL32 x
MULBY2	REAL32	VAL REAL32 x
NEXTAFTER	REAL32	VAL REAL32 x, y
NORMALISE	INT, INT, INT	VAL INT hi.in, lo.in
NOTFINITE	BOOL	VAL REAL32 x
ORDERED	BOOL	VAL REAL32 x, y
REAL32EQ	BOOL	VAL REAL32 x, y
REAL32GT	BOOL	VAL REAL32 x, y
REAL32OP	REAL32	VAL REAL32 x, VAL INT op, VAL REAL32 y

Function	Result(s)	Parameter specifiers
REAL32REM	REAL32	VAL REAL32 x, y
REAL64EQ	BOOL	VAL REAL64 x, y
REAL64GT	BOOL	VAL REAL64 x, y
REAL64OP	REAL64	VAL REAL64 x, VAL INT op, VAL REAL64 y
REAL64REM	REAL64	VAL REAL64 x, y
ROTATELEFT	INT	VAL INT argument, places
ROTATERIGHT	INT	VAL INT argument, places
SCALEB	REAL32	VAL REAL32 x, VAL INT n
SHIFTLEFT	INT, INT	VAL INT hi.in, lo.in, places
SHIFTRIGHT	INT, INT	VAL INT hi.in, lo.in, places
SQRT	REAL32	VAL REAL32 x

Dynamic code loading support procedures

Procedure	Parameter Specifiers
KERNEL.RUN	VAL []BYTE code, VAL INT entry.offset, []INT workspace, VAL INT no.of.parameters
LOAD.BYTE.VECTOR	INT here, []BYTE bytes
LOAD.INPUT.CHANNEL	INT here, CHAN OF ANY in
LOAD.INPUT.CHANNEL.VECTOR	INT here, []CHAN OF ANY in
LOAD.OUTPUT.CHANNEL	INT here, CHAN OF ANY out
LOAD.OUTPUT.CHANNEL.VECTOR	INT here, []CHAN OF ANY out

Transputer error flag manipulation

Procedure	Parameter Specifiers
CAUSEERROR	–
ASSERT	VAL BOOL test

Rescheduling priority process queue

Procedure	Parameter Specifiers
RESCHEDULE	–

inmos

Worldwide Headquarters

INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
UNITED KINGDOM
Telephone (0454) 616616
Fax (0454) 617910

Worldwide Business Centres

USA

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
2225 Executive Circle
PO Box 16000
Colorado Springs
Colorado 80935-6000
Telephone (719) 630 4000
Fax (719) 630 4325

SGS-THOMSON Microelectronics Inc.
Sales and Marketing Headquarters (USA)
1000 East Bell Road
Phoenix
Arizona 85022
Telephone (602) 867 6100
Fax (602) 867 6102

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
Lincoln North
55 Old Bedford Road
Lincoln
Massachusetts 01773
Telephone (617) 259 0300
Fax (617) 259 4420

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
9861 Broken Land Parkway
Suite 320
Columbia
Maryland 21045
Telephone (301) 995 6952
Fax (301) 290 7047

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
200 East Sandpointe
Suite 650
Santa Ana
California 92707
Telephone (714) 957 6018
Fax (714) 957 3281

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
2055 Gateway Place
Suite 300
San Jose
California 95110
Telephone (408) 452 9122
Fax (408) 452 0218

INMOS Business Centre
SGS-THOMSON Microelectronics Inc.
1310 Electronics Drive
Carrollton
Texas 75006
Telephone (214) 466 8844
Fax (214) 466 7352

ASIA PACIFIC

Japan

INMOS Business Centre
SGS-THOMSON Microelectronics K.K.
Nisseki Takanawa Building, 4th Floor
18–10 Takanawa 2-chome
Minato-ku
Tokyo 108
Telephone (03) 3280 4125
Fax (03) 3280 4131

Singapore

INMOS Business Centre
SGS-THOMSON Microelectronics Pte Ltd.
28 Ang Mo Kio Industrial Park 2
Singapore 2056
Telephone (65) 482 14 11
Fax (65) 482 02 40

EUROPE

United Kingdom

INMOS Business Centre
SGS-THOMSON Microelectronics Ltd.
Planar House
Parkway Globe Park
Marlow
Bucks SL7 1YL
Telephone (0628) 890 800
Fax (0628) 890 391

France

INMOS Business Centre
SGS-THOMSON Microelectronics SA
7 Avenue Gallieni
BP 93
94253 Gentilly Cedex
Telephone (1) 47 40 75 75
FAX (1) 47 40 79 10

West Germany

INMOS Business Centre
SGS-THOMSON Microelectronics GmbH
Bretonischer Ring 4
8011 Grasbrunn
Telephone (089) 46 00 60
Fax (089) 46 00 61 40

Italy

INMOS Business Centre
SGS-THOMSON Microelectronics SpA
V.le Milanofiori
Strada 4
Palazzo A/4/A
20090 Assago (MI)
Telephone (2) 89213 1
Fax (2) 8250449